

System Reliability Analysis with Dynamic Programming

T. Charng
DSN Engineering Section

System reliability analysis using the "Mission Success Diagram" method is presented in this article. In addition to the basic building blocks and reliability prediction equations, a BASIC computer program is written and tested with certain system configurations. This computer program can be modified for various system structures.

I. Introduction

Reliability, $R(t)$, is the probability that a device performs adequately over time interval $(0, t)$. In general, it is assumed that unless repair or replacement occurs, adequate performance at time t implies adequate performance during the interval $(0, t)$. The device under consideration may be an entire system, a subsystem containing one or more components, or a component. In principle, it is possible to break down the system into black boxes, with each black box being in one of two states: good or bad. Mathematical models of the system can then be abstracted from the physical processes and the theory of combinatorial probability utilized to predict the reliability of the system. The black boxes may be independent of, or dependent upon, each other.

When it is proposed to design a system to perform a complex and demanding task, it is assumed that the required investment will be justified according to the perfection by which the task is performed or by the large number of times which the system can do the job. This assumption cannot be justified when a system fails to perform upon demand or fails to perform repeatedly. Thus, it is not enough to show that a chasm can be spanned by a bridge; the bridge must continue to span the chasm for a long time to come while carrying a useful load.

Reliability is a problem at all levels. Reliability engineering (Refs. 1, 2) is concerned with the time degradation of materials, physical and electronic measurements, equipment design, process and system analysis, and synthesis.

To be of value, a prediction must be timely. The earlier a prediction has to be made about the unknown nature of a future event the more difficult it is to make a meaningful prediction.

It can be seen that the reliability of a device is known with certainty after it has been used in the field until it is worn out and its failure history has been faithfully recorded. Thus a fundamental limitation on reliability prediction is the ability to accumulate data of known validity for the new application. Another fundamental limitation is the complexity of prediction techniques. Very simple techniques omit a great deal of distinguishing detail and the prediction suffers inaccuracy. More detailed techniques can become so cumbersome that the prediction becomes costly and impractical.

II. Structural Function of a System

Suppose a system can be divided into n components. The performance of each component can be denoted by a random

variable X_i , which takes on the value $X_i = 1$ if the component performs satisfactorily for the desired time and $X_i = 0$ if the component fails during this time. In general then, X_i is a binary random variable defined by

$$X_i = \begin{cases} 1, & \text{if component } i \text{ performs satisfactorily during} \\ & \text{time } (0, t) \\ 0, & \text{if component } i \text{ fails during time } (0, t) \end{cases} \quad (1)$$

The performance of the system is measured by the binary random variable $\phi(X_1, X_2 \dots X_n)$, where

$$\phi(X_1, X_2 \dots X_n) = \begin{cases} 1, & \text{if system performs satisfactorily} \\ & \text{during time } (0, t) \\ 0, & \text{if system fails during time} \\ & (0, t) \end{cases} \quad (2)$$

The function ϕ is called the "structure function of the system" and is just a function of the n -component random variables.

Since the performance of each component in the system takes on the value 1 or 0, then the function ϕ is defined over 2^n points, with each point resulting in a 1 if the system performs satisfactorily and a zero if the system fails.

There are several important structure functions to consider, depending upon how the components are assembled. These are discussed in detail in the following.

III. Reliability of Series Systems

The series system is the simplest and most common of all configurations. For a series system, the system fails if any component of the system fails; i.e., it performs satisfactorily if and only if all the components perform satisfactorily. The structure function of a system containing n components is a binary random variable that takes on the value 1 or 0. Furthermore, the reliability of this system can be expressed as:

$$R = P\{\phi(X_1, X_2 \dots X_n) = 1\} \quad (3)$$

where P is the probability of success. When the usual times for conditional probability are employed,

$$R = P X_1 = 1 P\{X_2 = 1 | X_1 = 1\} P\{X_3 = 1 | X_1 = 1, X_2 = 1\} \dots P\{X_n = 1 | X_1 = 1, X_2 = 1, \dots X_{n-1} = 1\} \quad (4)$$

where $P\{X_2 = 1 | X_1 = 1\}$ is the probability that component 2 will perform successfully, given that component 1 performs

successfully. The performances of these components are then dependent, and the evaluation of the conditional probability is extremely difficult.

If, on the other hand, the performance characteristics of these components do not interact, then the components can be said to be independent. The expression for the reliability then simplifies and becomes:

$$R = P\{X_1 = 1\} P\{X_2 = 1\} \dots P\{X_n = 1\} \quad (5)$$

When components of a series system are assumed to be independent, it should be noted that the reliability is a function of the probability distribution of the X_i . This phenomenon is true for any system structure.

Unless otherwise specified it will be assumed throughout the remainder of this report that the component performances are independent. Hence, the probability distribution of the binary random variables X_i can be expressed as:

$$P\{X_i = 1\} = p_i \quad (6)$$

and

$$P\{X_i = 0\} = 1 - p_i \quad (7)$$

Thus for systems composed of independent components, the reliability becomes a function of the p_i ; that is,

$$R = R(p_1, p_2, \dots, p_n) = p_1 p_2 \dots p_n. \quad (8)$$

IV. Reliability of Parallel Systems

A parallel system of n components is defined to be a system that fails if all components fail, or alternatively, the system performs satisfactorily if at least one of the n components performs satisfactorily (with all n components operating simultaneously). This property of parallel systems is often called "redundancy"; there are alternative components, existing within the system, to keep the system operating successfully in case of failure of one or more components. The structure function for a parallel system is given by:

$$\phi(X_1, X_2, \dots, X_n) = 1 - (1 - X_1)(1 - X_2) \dots (1 - X_n). \quad (9)$$

The structure function takes on the value 1 if at least one of the X_i equals 1.

As previously indicated, for systems composed of independent components the reliability is given by:

$$R(p_1, p_2, \dots, p_n) = 1 - (1 - p_1)(1 - p_2) \dots (1 - p_n). \quad (10)$$

V. System Reliability Modeling

Building blocks is a method making use of the equations developed for redundancy to handle series, parallel combinations of equipments. For non-series parallel, series-parallel or complex configurations, use or repeated use of the following equation is required.

$$R = R(\text{if } X \text{ is good}) R_x + R(\text{if } X \text{ is bad}) Q_x \quad (11)$$

where R is reliability of the system, R_x is reliability of component X , and Q_x is unreliability of component $X = 1 - R_x$.

In order to construct a reliability model, procedures for one method are listed as follows:

- (1) Define what is required for mission success and translate this into a "mission success diagram."
- (2) Write the probability of success equation for the system.
- (3) Calculate the probability of success of the equipment in the system by utilizing one of the various reliability prediction techniques.
- (4) The probability numbers derived in Step 3 are inserted in the formula derived in Step 2.
- (5) A probability of success curve versus time can be plotted by taking several values of time for mission time, and evaluating the probability of system success by the above procedure for the several values of time chosen.
- (6) Additional steps in the analysis will depend upon the decisions that the analysis is intended to be optimized.

VI. Building Blocks of the System Reliability

Equations for basic building blocks of the probability of success are presented as follows for various "mission success diagrams." Each formula shown can be used as a building block to evaluate a more complex mission success diagram.

If there is only one component in the system, then the mission success diagram is as shown in Fig. 1(a). The reliability

for the system is obviously the probability of success of equipment A, or

$$R = p_A. \quad (12)$$

For two components in series, the mission success diagram is shown in Fig. 1(b), and the system reliability is

$$R = R(\text{if } A \text{ is good}) p_A + R(\text{if } A \text{ is bad}) (1 - p_A) \quad (13)$$

or

$$R = (p_B)(p_A) + (0)(1 - p_A) = p_A p_B \quad (14)$$

For two components in parallel, the mission success diagram is shown in Fig. 1(c).

The reliability of the system is

$$R = R(\text{if } A \text{ is good}) p_A + R(\text{if } A \text{ is bad}) (1 - p_A) \quad (13)$$

or

$$R = (1)(p_A) + (p_B)(1 - p_A) = p_A + p_B - p_A p_B \quad (15)$$

For a complex system with the mission success diagram as in Fig. 2(a).

The system reliability can be expressed as

$$R = R(\text{if } A \text{ is good}) p_A + R(\text{if } A \text{ is bad}) (1 - p_A). \quad (13)$$

When A is good, the system may be viewed as in Fig. 2(b), and

$$R(\text{if } A \text{ is good}) = p_{C1} + p_{C2} - p_{C1} p_{C2} \quad (16)$$

If A is bad, then the system may be viewed as in Fig. 2(c), and

$$R(\text{if } A \text{ is bad}) = p_{B1} p_{C1} + p_{B2} p_{C2} - p_{B1} p_{C1} p_{B2} p_{C2} \quad (17)$$

Thus the system reliability is:

$$\begin{aligned} R &= (p_{C1} + p_{C2} - p_{C1} p_{C2}) p_A \\ &\quad + (p_{B1} p_{C1} + p_{B2} p_{C2})(1 - p_A) \end{aligned} \quad (18)$$

VII. Computer Program

A computer program for the system reliability analysis written in the BASIC language utilizes dynamic programming technique (Refs. 3-5). In contrast to other optimization techniques (such as linear programming), a standard mathematical formulation does not exist. Dynamic programming starts with a small portion of the problem and finds the optimal solution for this smaller problem. It then gradually enlarges the problem, finding the current optimal solution from the previous one, until the original problem is solved in its entirety. Consequently, dynamic programming is a general strategy for optimization rather than a specific set of rules; the particular set of equations used must be developed to fit each problem.

The system reliability can be improved by providing redundancy in one or more of the components as mentioned before. If the probability of that component (N) functioning is P , then the probability that it will function with X units in parallel will be,

$$R(N, X) = P(N, X) \quad (19)$$

The probability that all components in the system will function will be the product of these probabilities for all components. The installation of these components will usually involve additional cost. The number of parallel units may be constrained by the size and weight of the system in addition to budgetary constraints.

In dynamic programming terminology, the STATE S of the system is the number of dollars available to a particular component to improve its reliability. The components will correspond to the STAGE N . The DECISION X will be the number of units of a component to be installed in parallel.

The recursive relation for this problem is:

$$f_N^* = OPT \{f_N(N, S, X)\} = R(N, X)f_{N-1}^* [S - M(N, X)] \quad (20)$$

where f^* is the optimum value determined by the S , N , and X .

The probabilities P and the cost M for each value of X are in arrays of P and M of the computer program.

In the case where the probability of each component with parallel installation is not determined experimentally, a similar recursive relation uses theoretical estimates for the probability of success instead of using tabulated probability values. The modified recursive relation uses theoretical estimates for the probability of success instead of tabulated probability values.

The modified recursive function is

$$\begin{aligned} f_N^* &= OPT \{f_N(N, S, X)\} \\ &= [1.0 - R(N, 1)] f_{N-1}^* [S - M(N, X)] \end{aligned} \quad (21)$$

where

$$R(N, 1) = 1.0 - P(N, 1)^x \quad (22)$$

The computer program listing is given in Appendix A. The description of the technique may be illustrated with the following two examples.

VIII. Numerical Examples

First let us consider the reliability problem with a system composed of four components as shown in Fig. 3. Each component has the probability of functioning with 1, 2, and 3 parallel units as given in Table 1. The cost of installing 1, 2, or 3 units in each of the components is tabulated in Table 2. The constraint is that no more than \$45K can be spent on the system. The question is how many parallel units in each component should be installed in order to maximize the system reliability?

The input information and computer printout is given in Appendix B. The solution of the problem is:

Total cost (in \$1000):	45
System reliability:	0.22
Redundancy of components:	2 units of component 1 2 units of component 2 2 units of component 3 1 unit of component 4

The second example, as shown in Fig. 4, is similar to the first example except that the probability of functioning for parallel components is not known. A theoretical estimate for the probability of success is expressed by Eq. (22). The recursive relation is given by Eq. (21).

With the input information given in Appendix C, the solution of a maximum of four parallel units for each component allowed is that:

Total cost (in \$1000):	45
System reliability:	0.28
Redundancy of components:	2 units of component 1 2 units of component 2 1 unit of component 3 2 units of component 4

IX. Summary

Building blocks and reliability equations based on the "mission success diagram" method is presented. Two numerical examples were solved with a Hewlett-Packard 2647A graphics terminal. The computer program is proved capable of analyzing system reliability by providing redundancy in

one or more of the components subject to size and weight of the system in addition to budgetary constraints.

This computer program provides a main framework based on the dynamic programming optimization technique. More generalized recursive relations or additional recursive relations may be required for different system configurations.

Acknowledgment

The author would like to acknowledge Dr. F. L. Lansing, who made a number of helpful suggestions in the study preparation.

References

1. Hillies, F. S., and Lieberman, G. J., *Introduction to Operations Research*, 3rd Edition, Holden-Day, Inc., San Francisco, CA, 1980.
2. Military Standardization Handbook, *Reliability Prediction of Electronic Equipment*, U.S. Dept. of Defense, 1978.
3. Bellman, R. E., and Dreyfus, S. E., *Applied Dynamic Programming*, Princeton University Press, Princeton, N.J., 1962.
4. Kuester, J. L., and Mize, J. H., *Optimization Techniques with Fortran*, McGraw-Hill Book Co., N.Y., 1973.
5. Rider, E., *General Computer Solution of Dynamic Programming Problems with Integer Restrictions*, M.S. thesis, Arizona State University, 1971.

Table 1. Probability of functioning for example 1

Component	Number of parallel units		
	1	2	3
1	0.60	0.75	0.85
2	0.40	0.65	0.80
3	0.70	0.90	0.95
4	0.50	0.60	0.80

Table 2. Costs of redundancy for example 1, in \$1000

Component	Number of parallel units		
	1	2	3
1	\$ 6	\$11	\$15
2	10	16	22
3	5	10	14
4	8	13	17

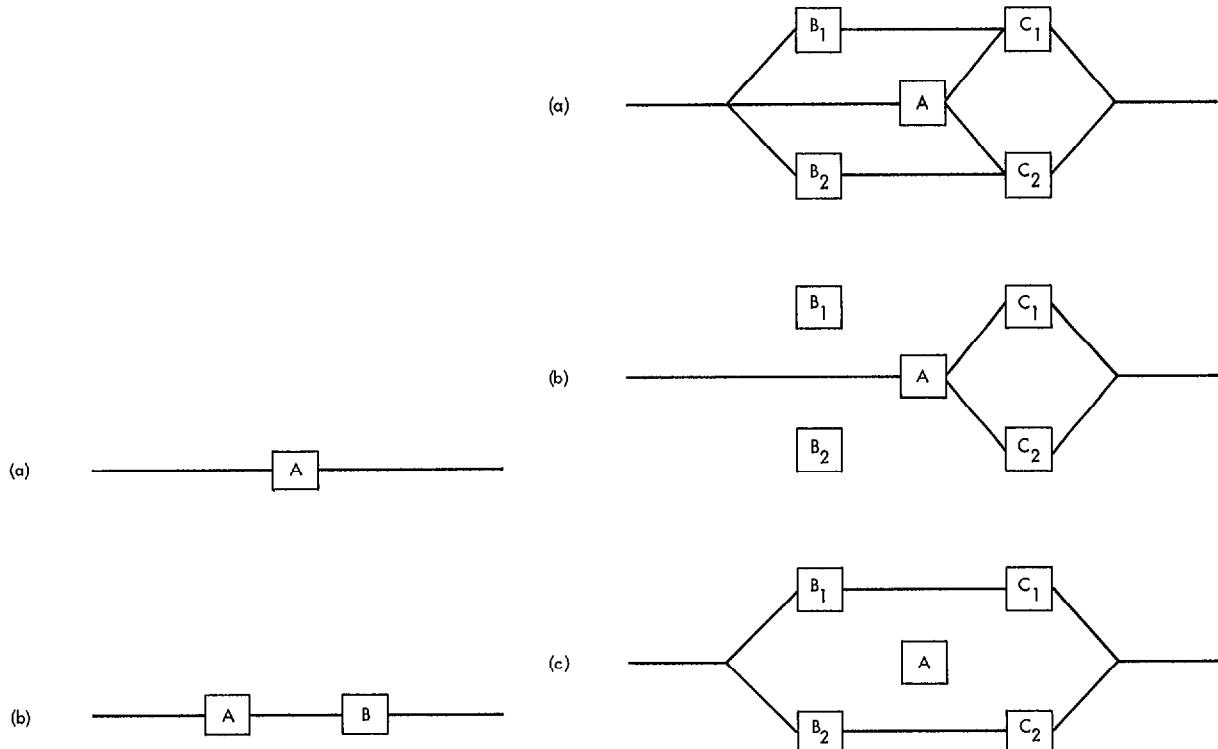


Fig. 2. A complex mission success diagram

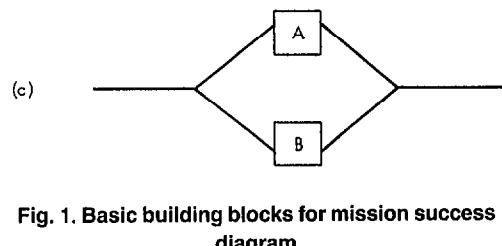


Fig. 1. Basic building blocks for mission success diagram

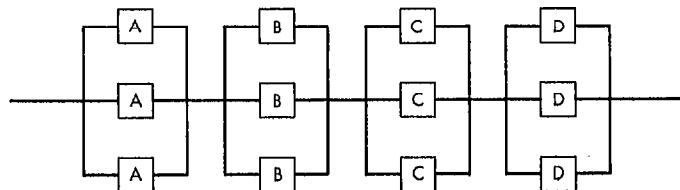


Fig. 3. Mission success diagram for example 1

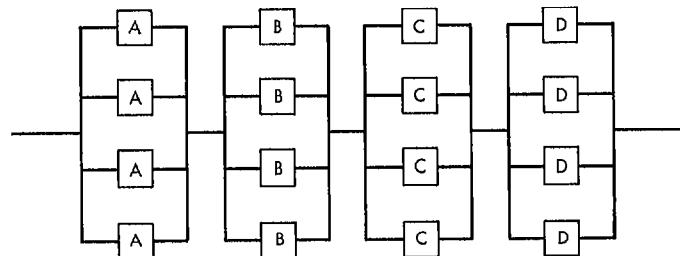


Fig. 4. Mission success diagram for example 2

Appendix A

Program Listing

```

50 REM..... REM
51 REM.. REM
52 REM.. GENERALIZED DYNAMIC PROGRAMMING PACKAGE .. REM
54 REM.. POSITIVE VALUED INTEGER SOLUTIONS ONLY .. REM
55 REM.. .. REM
56 REM.. T. CHARNG 1 OCTOBER 1981 .. REM
57 REM.. .. REM
70 REM..... REM
71 REM DIM Fsbest(Maxsze),Idbest(Maxsze),Fsopt(Maxsze) REM
72 REM DIM Kdec(Nstage),Kties(Nstage),Mtb(Nstage) REM
73 REM DIM Lps(Nstage),Mps(Nps),Lpd(Nstage),Mpd(Npd) REM
74 REM DIM P(Nstage,Ncolsp),C(Nrowsc,Ncolsc),M(Nstage,Ncolsm) REM
80 REM..... REM
81 DIM Fsbest(100),Idbest(100),Fsopt(100)
82 DIM Kdec(10),Kties(10),Mtb(10)
83 DIM Lps(10),Mps(20),Lpd(10),Mpd(20)
84 DIM P(10,10),C(10,10),M(10,10)
85 DIM Comment$(70)
90 REM..... REM
91 INTEGER Hs,Hd,Funct
99 REM..... DATA BLOCK ..... REM
100 REM ENTER DATA FUNCT, MIN, IOPT3, IOPT2, IOPT1
150 REM ENTER DATA NSTAGE, LOWD, MAXD, LOWS, MAXS, INC
200 REM ENTER DATA NCOLSP, NCOLSM, NPD, NPS, NROWSC, NCOLSC
250 REM ENTER DATA ((P(I,J), J=1,NCOLSP), I=1,NSTAGE)
350 REM ENTER DATA ((M(I,J), J=1,NCOLSM), I=1,NSTAGE)
400 REM ENTER DATA (MPD(I), I=1,NPD)
450 REM ENTER DATA (MPS(I), I=1,NPS)
500 REM ENTER DATA (MTB(I), I=1,NSTAGE)
550 REM ENTER DATA (FSOPT(I), I=1,NNS)
600 REM ENTER DATA (IDBEST(I), I=1,NNS)
900 REM ENTER COMMENTS
999 REM..... END OF DATA BLOCK ..... REM
1000 REM
1010 REM..... OPEN READ/PRINT FILES ..... REM
1012 Ki=5
1014 REM ..... ASSIGN "INPUT" TO #KI
1015 Ko=6
1020 ASSIGN "OUTPUT" TO #Ko
1050 REM..... PROGRAM STARTS ..... REM
1125 Maxsze=100
1140 READ Funct,Min,Iopt3,Iopt2,Iopt1
1145 Bigm=-9E+37
1150 IF Min>0 THEN Bigm=-Bigm
1160 IF Funct=4 THEN Iopt3=1
1170 IF Funct=5 THEN Iopt3=1
1172 PRINT #Ko;LIN(1);TAB(5);"PROBLEM TYPE" ;Funct
1173 PRINT #Ko;TAB(5);"MINIMIZATION OPTION" ;Min
1174 PRINT #Ko;TAB(5);"OUTPUT OPTION" ;Iopt3
1175 PRINT #Ko;TAB(5);"TIE-BREAKER OPTION" ;Iopt2
1176 PRINT #Ko;TAB(5);"STAGE 1 INPUT OPTION" ;Iopt1

```

```

1178 REM.....REM
1180 Incrmt=1
1190 READ Nstage,Lowd,Maxd,Lows,Maxs,Inc
1200 IF Inc>0 THEN Incrmt=Inc
1202 PRINT #Ko;LIN(1);TAB(5);"NUMBER OF STAGES" ;Nstage
1203 PRINT #Ko;TAB(5);"RANGE OF DECISION IS" ;Lowd;" TO ";Maxd
1204 PRINT #Ko;TAB(5);"RANGE OF STATES IS" ;Lows;" TO ";Maxs
1205 PRINT #Ko;TAB(5);"INCREMENT" ;Incrmt
1210 READ NcolsP,NcolsM,Npd,Nps,NrowsC,NcolsC
1211 IF Iopt3<0 THEN 1220
1212 PRINT #Ko;LIN(1);TAB(5);"NO. OF COLS IN 'P' MATRIX" ;NcolsP
1213 PRINT #Ko;TAB(5);"NO. OF COLS IN 'M' MATRIX" ;NcolsM
1214 PRINT #Ko;TAB(5);"NO. OF ELEMNT. IN 'MPD'" ;Npd
1215 PRINT #Ko;TAB(5);"NO. OF ELEMNT. IN 'MPS'" ;Nps
1216 PRINT #Ko;TAB(5);"NO. OF ROWS IN 'C' MATRIX" ;NrowsC
1217 PRINT #Ko;TAB(5);"NO. OF COLS IN 'C' MATRIX" ;NcolsC
1219 REM.....REM
1220 IF NcolsP<=0 THEN 1280
1225 FOR I=1 TO Nstage
1230 FOR J=1 TO NcolsP
1235 READ P(I,J)
1240 NEXT J
1245 NEXT I
1247 IF Iopt3<0 THEN 1280
1250 PRINT #Ko;LIN(1)
1251 FOR I=1 TO Nstage
1252 IF NcolsP>=4 THEN 1260
1253 FOR J=1 TO NcolsP
1254 PRINT #Ko;TAB(5);P(";I;";"J")=";P(I,J);
1255 NEXT J
1256 GOTO 1274
1260 FOR J=1 TO NcolsP STEP 4
1261 PRINT #Ko;TAB(5);P(";I;";"J+0")=";P(I,J+0);
1262 PRINT #Ko;TAB(5);P(";I;";"J+1")=";P(I,J+1);
1263 PRINT #Ko;TAB(5);P(";I;";"J+2")=";P(I,J+2);
1264 PRINT #Ko;TAB(5);P(";I;";"J+3")=";P(I,J+3)
1270 NEXT J
1274 PRINT #Ko;LIN(1)
1275 NEXT I
1280 IF NrowsC<=0 THEN 1350
1282 IF NcolsC<=0 THEN 1350
1285 FOR I=1 TO NrowsC
1286 FOR J=1 TO NcolsC
1290 READ C(I,J)
1295 NEXT J
1296 NEXT I
1300 IF Iopt3<0 THEN 1350
1305 PRINT #Ko;LIN(1)
1310 FOR I=1 TO NrowsC
1312 IF NcolsM>=4 THEN 1322
1314 FOR J=1 TO NcolsC
1316 PRINT #Ko;TAB(5);C(";I;";"J")=";C(I,J);
1318 NEXT J
1320 GOTO 1335

```

```

1322 FOR J=1 TO NcolsC STEP 4
1325 PRINT #Ko;TAB(5);"C(";I;",";J+0;")=";C(I,J+0);
1326 PRINT #Ko;TAB(5);"C(";I;",";J+1;")=";C(I,J+1);
1327 PRINT #Ko;TAB(5);"C(";I;",";J+2;")=";C(I,J+2);
1328 PRINT #Ko;TAB(5);"C(";I;",";J+3;")=";C(I,J+3)
1330 NEXT J
1335 PRINT #Ko;LIN(1)
1340 NEXT I
1345 REM. ....
1350 IF NcolsM<=0 THEN 1410
1355 FOR I=1 TO Nstage
1357 FOR J=1 TO NcolsM
1360 READ M(I,J)
1365 NEXT J
1367 NEXT I
1368 IF Iopt3<0 THEN 1410
1369 PRINT #Ko;LIN(1)
1370 FOR I=1 TO Nstage
1371 IF NcolsM>=4 THEN 1380
1372 FOR J=1 TO NcolsM
1375 PRINT #Ko;TAB(5);"M(";I;",";J;")=";M(I,J);
1377 NEXT J
1378 GOTO 1395
1380 FOR J=1 TO NcolsM STEP 4
1382 PRINT #Ko;TAB(5);"M(";I;",";J+0;")=";M(I,J+0);
1383 PRINT #Ko;TAB(5);"M(";I;",";J+1;")=";M(I,J+1);
1384 PRINT #Ko;TAB(5);"M(";I;",";J+2;")=";M(I,J+2);
1385 PRINT #Ko;TAB(5);"M(";I;",";J+3;")=";M(I,J+3)
1390 NEXT J
1395 PRINT #Ko;LIN(1)
1400 NEXT I
1405 REM. ....
1410 IF Npd<=0 THEN 1610
1415 FOR I=1 TO Npd
1417 READ MpD(I)
1420 NEXT I
1430 IF Iopt3<0 THEN 1610
1432 PRINT #Ko;LIN(1)
1435 FOR I=1 TO Npd
1437 PRINT #Ko;TAB(5);"MPD(";I;")=";MpD(I);
1440 NEXT I
1450 N=1
1460 KMM=1
1470 FOR I=1 TO Npd
1480 IF MpD(KMM)<0 THEN 1520
1490 Lpd(N)=KMM
1500 KMM=KMM+1
1510 GOTO 1540
1520 Lpd(N)=-(KMM+1)
1530 KMM=KMM-MpD(KMM)
1540 IF KMM<=Npd THEN 1560
1550 PRINT #Ko;"$$$$$$ ERROR CONDITION 1$$$$$"
1555 GOTO 60000
1560 N=N+1

```

```

1570 IF N>Nstage THEN 1610
1580 Kmm=Kmm+1
1590 NEXT I
1600 REM..... REM
1610 IF Nps<=0 THEN 1800
1620 FOR I=1 TO Nps
1622 READ Mps(I)
1624 NEXT I
1628 IF Iopt3<0 THEN 1800
1629 PRINT #Ko,LIN(1)
1630 FOR I=1 TO Nps
1635 PRINT #Ko,TAB(5); "MPS(";I;")=";Mps(I);
1637 NEXT I
1640 N=1
1650 Kmm=1
1660 FOR I=1 TO Nps
1670 IF Mps(Kmm)<0 THEN 1710
1680 Lps(N)=Kmm
1690 Kmm=Kmm+1
1700 GOTO 1730
1710 Lps(N)=-(Kmm+1)
1720 Kmm=Kmm-Mps(Kmm)
1730 IF Kmm<=Nps THEN 1750
1740 PRINT #Ko;"$$$$$$ ERROR CONDITION 2 $$$$$"
1745 GOTO 60000
1750 N=N+1
1760 IF N>Nstage THEN 1800
1770 Kmm=Kmm+1
1780 NEXT I
1790 REM..... REM
1800 IF Iopt2<=1 THEN 1850
1802 FOR I=1 TO Nstage
1804 READ MtB(I)
1806 NEXT I
1840 GOTO 1880
1850 FOR I=1 TO Nstage
1860 MtB(I)=Iopt2
1870 NEXT I
1880 IF Iopt3<0 THEN 1890
1882 PRINT #Ko,LIN(1)
1884 FOR I=1 TO Nstage
1886 PRINT #Ko,TAB(5); "MTB(";I;")=";MtB(I)
1888 NEXT I
1890 Nns=(Maxs-Lows+Incrmt)/Incrmt
1900 IF Nns<=Maxsze THEN 1920
1910 PRINT #Ko;"$$$$$$ ERROR CONDITION 3 $$$$$"
1915 GOTO 60000
1920 IF Iopt1<=0 THEN 1940
1922 FOR I=1 TO Nns
1924 READ Fsept(I)
1926 NEXT I
1928 FOR I=1 TO Nns
1930 READ Idbest(I)
1932 NEXT I

```

```

1940 COMMAND "M F H HP-IB#1"
1942 PRINT #Ko;LIN(2)
1950 READ Comment$
1952 IF Comment$="END" THEN 2000
1954 PRINT #Ko;TAB(5);Comment$
1956 GOTO 1950
1960 FOR I=1 TO Nns
1970 READ Idbest(I)
1980 NEXT I
2000 REM..... END OF INPUT SECTION ..... REM
2010 REM
2020 REM ..... OPEN TAPE FILES ..... REM
2031 Ntp1=1
2032 ASSIGN "TAPE1" TO #Ntp1
2033 REM
2034 COMMAND "RE TAPE1"
2035 Ntp2=2
2036 ASSIGN "TAPE2" TO #Ntp2
2037 COMMAND "RE TAPE2"
2039 REM..... REM
2040 Moetp=-1
2050 Mflag=0
2055 Kfile=0
2056 REM..... REM
2060 PRINT #Ko;LIN(3)
2065 PRINT #Ko;TAB(5); "STAGE", "STATE", "OPT DECISION", "OPT VALUEN"
2066 PRINT #Ko;TAB(5); "----", "----", "----", "----", "----"
2067 PRINT #Ntp2;TAB(5); "STAGE", "STATE", "DECISION", "VALUE", "TIED"
2070 PRINT #Ntp2;TAB(5); "----", "----", "----", "----", "----"
2075 REM..... REM
2080 N1=1
2090 IF Iopti<=0 THEN 2240
2095 Ns=Lows-Incrmt
2100 FOR K=1 TO Nns
2110 Ns=Ns+K*Incrmt
2120 PRINT #Ko;TAB(5);N1,Ns,Idbest(K),Fsopt(K)
2130 NEXT K
2171 FOR L=1 TO Nns
2180 PRINT #Ntp1;Idbest(L);
2190 NEXT L
2220 Command" M F H TAPE"
2230 N1=2
2235 REM
2240 REM..... START MAIN LOOP ..... REM
2245 REM
2250 FOR N=N1 TO Nstage
2260 REM..... INITALIZE FSBEST(I) & IDBEST(I) ..... REM
2270 FOR K=1 TO Nns
2280 Fsbest(K)=Bigm
2290 Idbest(K)=8888
2300 NEXT K
2310 REM..... INITILIZE THE STATE VARIABLE 'NS' ..... REM
2320 IF Nps>0 THEN 2410
2330 Ns=Lows
2340 Hs=Maxs

```

```

2350 IF N<>Nstage THEN 2570
2360 IF Nps=-1 THEN Ns=Maxs
2390 IF Nps=-2 THEN Hs=Lows
2400 GOTO 2570
2410 Locs=Lps(N) ! 150
2420 IF Locs<0 THEN 2500
2430 Ns=Mps(Locs)
2440 IF Ns>Maxs THEN 2455
2450 IF Ns>=Lows THEN 2460
2455 PRINT #Ko;"$$$$$$ ERROR CONDITION 5    $$$$$"
2458 GOTO 60000
2460 Hs=Mps(Locs+1)
2470 IF Hs>Maxs THEN 2490
2480 IF Hs>=Ns THEN 2570
2490 PRINT #Ko;"$$$$$$ ERROR CONDITION 6    $$$$$"
2496 GOTO 60000
2500 Locs=-Locs ! 175
2510 Ns=Mps(Locs)
2520 IF Ns>Maxs THEN 2535
2530 IF Ns>=Lows THEN 2540
2535 PRINT #Ko;"$$$$$$ ERROR CONDITION 7    $$$$$"
2536 GOTO 60000
2540 K=Locs-1
2550 Lshigh=K-Mps(K)
2570 REM..... INITIALIZE THE DECISION VARIABLE 'ND' . . . . . REM
2571 I=(Ns-Lows+Incrmt)/Incrmt ! 200
2580 Idbest(I)=8888
2590 IF Npd>0 THEN 2630
2600 Nd=Loud
2610 Hd=Maxd
2620 GOTO 2790
2630 Locd=Lpd(N) ! 250
2640 IF Locd<0 THEN 2720
2650 Nd=Mpd(Locd)
2660 IF Nd>Maxd THEN 2675
2670 IF Nd>=Loud THEN 2680
2675 PRINT #Ko;"$$$$$$ ERROR CONDITION 9    $$$$$"
2676 GOTO 60000
2680 Hd=Mpd(Locd+1)
2690 IF Hd>Maxd THEN 2710
2700 IF Hd>=Nd THEN 2790
2710 PRINT #Ko;"$$$$$$ ERROR CONDITION 10    $$$$$"
2715 GOTO 60000
2720 Locd=-Locd ! 275
2730 Nd=Mpd(Locd)
2740 IF Nd>Maxd THEN 2754
2750 IF Nd>=Loud THEN 2760
2754 PRINT #Ko;"$$$$$$ ERROR CONDITION 11    $$$$$"
2755 GOTO 60000
2760 K=Locd-1
2770 Ldhigh=K-Mpd(K)
2790 REM..... USE THE RECURSIVE FUNCTIONS ..... REM
2791 IF Funct<=0 THEN 2800 ! 300
2792 IF Funct<=9 THEN 2805

```

```

2800 PRINT #Ko;"$$$$$$ ERROR CONDITION 14 $$$$$"
2802 GOTO 60000
2805 Kn=N ! 400
2806 Ks=Ns
2807 Kx=Nd
2810 ON Funct GOSUB 5520,5660,5760,5800,5930,6310,6430,6570,6760
2815 REM
2820 Itied=0 ! 310
2830 IF FnSxn=Fsbest(I) THEN Itied=1
2835 PRINT #Ntp2,TAB(5);N,Ns,Nd,FnSxn
2840 IF Min>0 THEN 3080
2850 REM..... MAXIMIZING LOGIC ..... REM
2860 IF MtB(N)<=0 THEN 2890
2870 IF FnSxn>=Fsbest(I) THEN 2920
2880 GOTO 2950
2890 IF FnSxn<=Fsbest(I) THEN 2950 ! 325
2900 REM..... UPDATE VALUE AND DECISION..... REM
2920 Fsbest(I)=FnSxn ! 340
2930 Idbest(I)=Nd
2940 REM..... INCREMENT THE DECISION VARIABLE 'ND' .. REM
2950 IF Itied=0 THEN 2980 ! 350
2960 Ix=Idbest(I)
2970 Idbest(I)=-ABS(Ix)
2980 IF Npd<=0 THEN 3050 ! 355
2990 IF Lpd(N)>=1 THEN 3050
3000 Locd=Locd+1
3010 IF Locd>Ldhhigh THEN 3180
3020 Nd=Mpd(Locd)
3030 IF Nd<=Maxd THEN 2790
3034 IF Nd>=Lowd THEN 2790
3040 PRINT #Ko;"$$$$$$ ERROR CONDITION 12 $$$$$"
3045 GOTO 60000
3050 Nd=Nd+Incrmt ! 375
3060 IF Nd<=Hd THEN 2790
3070 GOTO 3180
3080 REM..... MINIMIZING LOGIC ..... REM
3090 IF MtB(N)<=0 THEN 3120 ! 380
3100 IF FnSxn<=Fsbest(I) THEN 2920
3110 GOTO 2950
3120 IF FnSxn>=Fsbest(I) THEN 2920 ! 390
3130 GOTO 2950
3140 IF Iopt3>0 THEN 3220 ! 500
3150 IF N>Nstage THEN 3310
3220 REM..... OUTPUT OPTIONAL EXCEPT FOR LAST STAGE .. REM
3221 Ix=Idbest(I)
3230 Itied$="" "
3240 IF Ix>=0 THEN 3280
3250 Ix=-Ix
3260 Itied$="**"
3270 MFlag=1
3280 PRINT #Ko;TAB(5);N,Ns,Ix,Fsbest(I);Itied$ ! 505
3290 IF N=Nstage THEN 3580
3310 REM..... INCREMENT THE STATE VARIABLE 'NS' ..... REM
3320 PRINT #Ntp2,LIN(1)

```

```

3321 IF Nps<=0 THEN 3390 ! 510
3322 IF Lps(N)>=1 THEN 3390
3330 Locs=Locs+1
3340 IF Locs>Lshigh THEN 3410
3350 Ns=Mps(Locs)
3360 IF Ns>Maxs THEN 3380
3370 IF Ns<Lowest THEN 3380
3375 GOTO 2570
3380 PRINT #Ko;"$$$$$$ ERROR CONDITION 8 $$$$$"
3382 GOTO 60000
3390 Ns=Ns+Incrmt ! 515
3400 IF Ns<=Ns THEN 2570
3410 IF Iopt3>0 THEN PRINT #6;LIN(1) ! 525
3420 IF N=Nstage THEN 4430
3430 REM..... SAVE CURRENT STAGE DECISION ON TAPE1 . . . REM
3471 FOR L=1 TO Nns
3480 PRINT #Ntpi;Idbest(L);
3490 NEXT L
3515 COMMAND "M F H TAPE1"
3530 REM..... UPDATE FSOPT(I) FOR USE AT NEXT STAGE . . . REM
3531 FOR K=1 TO Nns ! 540
3540 Fsopt(K)=Fsbest(K)
3550 NEXT K
3560 GOTO 4430
3580 REM..... RECOVER AND OUTPUT OF OPTIMAL DECISIONS REM
3581 N2=Nstage ! 600
3590 Js=Ns
3600 M2=1
3610 Kdec(M2)=Ix
3620 Nty=0
3630 IF Idbest(I)>=0 THEN 3660
3640 Nty=Nty+1
3650 Kties(Nty)=N2
3655 REM
3660 IF N2=1 THEN 4230 ! 625
3670 Jx=Kdec(M2)
3680 IF Jx=8888 THEN 3310
3690 IF Funct<=0 THEN 3310
3700 IF Funct>9 THEN 3310
3710 Kn=N2
3720 Ks=Js
3730 Kx=Jx
3740 REM..... LOOKUP THE RECOVERY FUNCTION . . . . . REM
3750 ON Funct GOSUB 5240,5210,5210,3310,3310,5240,5360,5330,5270
3760 Js=News
3770 IF Js=8888 THEN 3310
3780 IF N2<>Nstage THEN 3870 ! 705
3790 IF Moetp>0 THEN 3910
3830 REM..... PUT TAPE IN READING MODE . . . . . REM
3840 COMMAND "F F -1 TAPE1"
3845 Moetp=-Moetp
3850 GOTO 3940
3860 REM..... BACKSPACE TAPE TO POSITION IN FRONT OF NEXT STAGE . . . REM
3870 COMMAND "F F -1 TAPE1" ! 610

```

```

3900 REM..... REM
3910 COMMAND "F F -1 TAPE1" ! 620
3940 REM..... READ OPT DECISION OF PREVIOUS STAGE ...REM
3950 FOR L=1 TO Nns ! 623
3960 READ #Ntp1;Idbest(L)
3970 NEXT L
3980 Kss=(Js-Lows+Incrmt)/Incrmt
3990 N2=N2-1
4000 M2=M2+1
4020 Kdec(M2)=Idbest(Kss)
4030 IF Kdec(M2)>=0 THEN 3660
4040 Kdec(M2)=-Kdec(M2)
4070 GOTO 3640
4220 REM..... REM
4230 N3=Nstage+1 ! 850
4231 PRINT #Ko;LIN(2)
4235 PRINT #Ko;TAB(15);"STAGE","DECISION"
4236 PRINT #Ko;TAB(15);"-----","-----"
4240 FOR K=1 TO Nstage
4250 PRINT #Ko;TAB(15);N3-K,Kdec(K)
4260 NEXT K
4270 PRINT #Ko;LIN(2)
4280 IF Nty=0 THEN 4340
4285 PRINT #Ko;TAB(5);"ALTERNATE OPTIMAL DECISIONS EXIST AT STAGE'S:"
4290 FOR K=1 TO Nty
4300 PRINT #Ko;TAB(5);Kties(K)
4310 NEXT K
4320 PRINT #Ko;LIN(1)
4340 REM..... REPOSITION TAPE .....REM
4395 COMMAND "F E D L"
4400 COMMAND "F F -1 TAPE1"
4410 GOTO 3310
4420 REM
4430 PRINT #Ntp2;LIN(1)
4435 NEXT N
4440 REM..... END OF THE MAIN LOOP FOR EACH STAGE ...REM
4450 IF Mflag<=0 THEN 60000
4460 PRINT #6;TAB(5);" ** AFTER THE OPTIMUM DECISION INDICATES THAT"
4470 PRINT #6;TAB(5);" ALTERNATE OPTIMAL DECISIONS EXIST."
4480 PRINT #6;TAB(5);" THE TIE-BREAKER OPTION CONTROLS SELECTION"
4490 PRINT #6;TAB(5);" OF THE LOWEST OR HIGHEST DECISION."
4540 GOTO 60000
4550 REM 1 2
4560 REM..... END OF PROGRAM .....REM
4570 REM LIST 4340
4600 REM..... FUNCTION LOCATE (IJ) .....REM
4610 Locat=(IJ-Lows+Incrmt)/Incrmt
4620 RETURN
4700 REM..... FUNCTION LOCATD (MN,MD) .....REM
4702 IF Npd>0 THEN 4708
4704 L1=Loud
4706 GOTO 4714
4708 IF Lpd(Mn)<=0 THEN 4718

```

```

4710 Kk=Lpd(Mn)
4712 L1=Mpd(Kk)
4714 Locatd=(Md-L1+Incrmt)/Incrmt
4716 RETURN
4718 Kk=-Lpd(Mn)
4720 Locatd=0
4722 Jdec=Mpd(Kk)
4724 Locatd=Locatd+1
4726 IF Jdec=Md THEN 4716
4728 Kk=Kk+1
4730 IF Kk<=Ldhigh THEN 4722
4732 PRINT #Ko;"$$$$$$ ERROR CONDITION 13 $$$$$"
4734 GOTO 60000
4800 REM..... FUNCTION FSTAR (NEW,VL,VH,V1) .... REM
4802 IF New<Lows THEN 4814
4804 IF New>Maxs THEN 4818
4806 IF N=1 THEN 4822
4808 Kk=(New-Lows+Incrmt)/Incrmt
4810 Fstar=Fsopt(Kk)
4812 RETURN
4814 Fstar=V1
4816 RETURN
4818 Fstar=Vh
4820 RETURN
4822 Fstar=V1
4824 RETURN
38600 Jx=Kdec(M2)
60000 REM..... REM
60010 COMMAND "RE TAPE1"
60011 COMMAND "RE TAPE2"
60020 COMMAND "M F H HP-IB#1"
60030 PRINT #Ko;LIN(3)
60040 PRINT #Ko; "", "*****"
60060 PRINT #Ko; "", "*"
60070 PRINT #Ko; "", "*"
60080 PRINT #Ko; "", "*"
60090 PRINT #Ko; "", "*****"
60100 END

```

```
5270 REM..... FUNCTION NEWS3 (KN,KS,KX) .....REM
5280 IF NcolsM>1 THEN 5305
5290 News=Ks-(Kx)*M(Kn,1)
5300 RETURN
5305 Mn=Kn
5306 Md=Kx
5308 GOSUB 4700
5310 News=Ks-M(Kn,Locatd)
5320 RETURN
6760 REM..... FUNCTION RECUR9 (KN,KS,KX) .....REM
6770 REM..... GENERAL SOLUTION FOR SYSTEM RELIABILITY .....REM
6800 IF NcolsP=1 THEN 6840
6802 Mn=Kn
6804 Md=Kx
6810 GOSUB 4700
6820 Fnexn=P(Kn,Locatd)
6830 GOTO 6850
6840 Fnexn=1-(1-P(Kn,1))^Kx
6850 Vl=BigM
6860 Vh=BigM
6870 Vl=1
6875 GOSUB 5280
6877 New=News
6880 GOSUB 4800
6890 Fnexn=Fnexn*Fstar
6900 RETURN
```

Appendix B

Example 1 Inputs and Result

```
7000 REM.....INPUT DATA.....REM
7100 REM ENTER DATA FUNCT, MIN, IOPT3, IOPT2, IOPT1
7110 DATA 9, 0, 0, 0, 0
7150 REM ENTER DATA NSTAGE, LOWD, MAXD, LOWEST, MAXS, INC
7151 DATA 4, 1, 3, 0, 45, 1
7200 REM ENTER DATA NCOLSP, NCOLSM, NPD, NPS, NROWSC, NCOLSC
7210 DATA 3, 3, 0, -1, 0, 0
7250 REM ENTER DATA ( (P(I,J), J=1,NCOLSP), I=1,NSTAGE )
7251 DATA .60, .75, .85
7252 DATA .40, .65, .80
7253 DATA .70, .90, .95
7254 DATA .50, .60, .80
7350 REM ENTER DATA ( (M(I,J), J=1,NCOLSM), I=1,NSTAGE)
7351 DATA 6, 11, 15
7352 DATA 10, 16, 22
7353 DATA 5, 10, 14
7354 DATA 8, 13, 17
7400 REM ENTER COMMENTS
7410 DATA "TYPE 9 EXAMPLE 1"
7411 DATA "
7412 DATA "
7420 DATA "STAGE:      COMPONENTS OF THE SYSTEM"
7421 DATA "STATE:      UNALLOCATED RESOURCES (DOLLARS)"
7422 DATA "DECISION:   NUMBER OF PARALLEL UNITS OF THE COMPONENT INSTALLED"
7423 DATA "VALUE:      PROBABILITY OF THE SYSTEM WILL FUNCTION SUCCESS"
7424 DATA "END"
```

TYPE 9 EXAMPLE 1

STAGE: COMPONENTS OF THE SYSTEM

STATE: UNALLOCATED RESOURCES (DOLLARS)

DECISION: NUMBER OF PARALLEL UNITS OF THE COMPONENT INSTALLED

VALUE: PROBABILITY OF THE SYSTEM WILL FUNCTION SUCCESS

STAGE	STATE	OPT DECISION	OPT VALUE
4	45	1	219375

STAGE	DECISION
4	1
3	2
2	2
1	2

Appendix C

Example 2 Inputs and Result

```
2000 REM ENTER DATA FUNCT, MIN, IOPT3, IOPT2, IOPT1
2001 DATA 9, 0, 0, 0, 0
2002 REM ENTER DATA NSTAGE, LOWD, MAXD, LOWEST, MAXS, INC
2003 DATA 4, 1, 4, 0, 45, 1
2004 REM ENTER DATA NCOLSP, NCOLSM, NPD, NPS, NROWSC, NCOLSC
2005 DATA 1, 4, 0,-1, 0, 0
2006 REM ENTER DATA ( (P(I,J), J=1,NCOLSP), I=1,NSTAGE )
2007 DATA .60
2008 DATA .40
2009 DATA .70
2010 DATA .50
2011 REM ENTER DATA ( (M(I,J), J=1,NCOLSM), I=1,NSTAGE )
2012 DATA 6, 11, 15, 18
2013 DATA 10, 16, 22, 27
2014 DATA 5, 10, 14, 17
2015 DATA 8, 13, 17, 20
2016 REM ENTER DATA (MPD(I), I=1,NPD)
2017 REM ENTER DATA (MPS(I), I=1,NPS)
2019 REM ENTER DATA (MTB(I), I=1,NSTAGE)
2020 DATA "TYPE 9 EXAMPLE 2"
2021 DATA "
2022 DATA "
2023 DATA "STAGE:      COMPONENTS OF THE SYSTEM"
2024 DATA "STATE:      UNALLOCATED RESOURCES (DOLLARS)"
2025 DATA "DECISION:   NUMBER OF PARALLEL UNITS OF THE COMPONENT INSTALLED"
2026 DATA "VALUE:      PROBABILITY OF THE SYSTEM WILL FUNCTION SUCCESS"
2027 DATA "END"
```

TYPE 9 EXAMPLE 2

STAGE: COMPONENTS OF THE SYSTEM

STATE: UNALLOCATED RESOURCES (DOLLARS)

DECISION: NUMBER OF PARALLEL UNITS OF THE COMPONENT INSTALLED

VALUE: PROBABILITY OF THE SYSTEM WILL FUNCTION SUCCESS

STAGE	STATE	OPT DECISION	OPT VALUE
4	45	2	.28224

STAGE	DECISION
4	2
3	1
2	2
1	2